CENTRE DE RECHERCHE SUR LES TRANSPORTS
(C.R.T.)
CENTRE FOR RESEARCH ON TRANSPORTATION

UNIVERSITÉ DE MONTRÉAL
ÉCOLE DES HAUTES ÉTUDES COMMERCIALES
ÉCOLE POLYTECHNIQUE

CENTRE DE RECHERCHE SUR LES TRANSPORTS
UNIVERSITÉ DE MONTRÉAL
C.P. 6128, SUCCURSALE A
MONTRÉAL
CANADA H3C 3J7

TÉLÉPHONE: (514) 343-7575
TÉLEX: (21) 05-561325
EMAIL: crt@crt.umontreal.ca
TÉLÉCOPIEUR (FAX): (514) 343-7121

CRT-797

A Tabu Search Procedure for Multicommodity Location/Allocation with Balancing Requirements

by
Teodor G. Crainic[1,2]
Michel Gendreau[1]
Patrick Soriano[1]
Michel Toulouse[1]

[1] Centre de recherche sur les transports, Université de Montréal, C.P. 6128, succursale A, Montréal, Québec, CANADA H3C 3J7.
[2] Département de sciences administratives, Université du Québec à Montréal, C.P. 6192, succursale A, Montréal, Québec, CANADA H3C 4R2.

Centre de recherche sur les transports — Publication #797

September 1991

92-00712/

Dépôt légal – 4ème trimestre 1991

Abstract

We propose a Tabu Search heuristic for the location/allocation problem with balancing requirements. This problem typically arises in the context of the medium term management of a fleet of containers of multiple types, where container depots have to be selected, the assignment of customers to depots has to be established for each type of container, and the interdepot container traffic has to be planned to account for differences in supplies and demands in various zones of the geographical territory served by a container shipping company. It is modeled as a mixed integer program, which combines zero-one location variables and a multicommodity network flow structure.

Extensive computational results on a set of benchmark problems and comparisons with an efficient dual ascent procedure are reported. These show that Tabu Search is a competitive approach for this class of problems.

Keywords: Tabu Search; Multicommodity location/allocation problems; Empty flow balancing; Container transportation.

- - - - - - - - - - - -

Résumé

Nous décrivons une heuristique de recherche avec tabous pour le problème de localisation/allocation avec flots d'équilibrage. Ce problème se pose dans le contexte de la gestion à moyen-terme d'une flotte de conteneurs de plusieurs types. Il consiste à choisir les dépôts de conteneurs à utiliser, à établir l'affectation des clients aux dépôts pour chaque type de conteneurs, et à planifier les flots de conteneurs vides entre les dépôts de façon à équilibrer les différences entre l'offre et la demande de conteneurs dans les diverses régions du territoire desservi par l'entreprise de transport par conteneurs. Le problème est modélisé comme un programme mixte en nombres entiers dans lequel on retrouve des variables de localisation binaires et une structure de flots de plusieurs produits sur un réseau.

Nous présentons des résultats numériques pour plusieurs problèmes et nous les comparons à ceux obtenus par une procédure d'ascension duale efficace. Ces résultats montrent que l'approche de recherche avec tabous est compétitive pour cette classe de problèmes.

Mots-clés: recherche avec tabous; problèmes de localisation/allocation à plusieurs produits; flots d'équilibrage; transport par conteneurs.

Introduction

These last few years have witnessed the emergence of a whole new class of optimization algorithms based on the Tabu Search technique. Tabu Search is commonly referred to as a *metaheuristic*, since it is used to explore the solution space of a given problem, iteratively, with another (or a set of other) local heuristic search procedure(s) being normally used at each step. Initially proposed by Glover (1986), and independently by Hansen and Jaumard (1987), Tabu Search is continuously enhanced (see Glover, 1989, 1990, and de Werra and Hertz, 1989, for the basic ingredients of the method), and applied to an increasingly wide spectrum of problems. Particularly impressive results have been obtained for hard combinatorial problems, such as Quadratic Programming (Skorin-Kapov, 1990), Vehicle Routing Problems (Gendreau, Hertz and Laporte, 1991) and Flow Shop Sequencing (Widmer and Hertz, 1989).

Significantly less attention has been directed towards problems which display not only combinatorial characteristics, but also a strong "continous" component. In particular, multicommodity network design problems, with, eventually, additional constraints, form such a class of models which, we believe, is a prime candidate for the successful application of Tabu Search based algorithms.

Such a study presents a double interest. First, these problems are "difficult" to solve, in the sense that traditional heuristics generally fail to produce optimal solutions and display significant gaps (especially when additionnal constraints are imposed on the multicommodity structure – see Balakrishnan, Magnanti and Wong, 1989), which implies that integer programming methods have to be applied. Second, the analysis of

the performance of Tabu Search procedures in the context of problems displaying both combinatorial characteristics and a significant number of continuous variables, may help gain significant insights into the behaviour of the Tabu Search methodology and into ways to enhance it.

For our part, we undertake this study by focusing on a particular case of this class of problems, namely the location/allocation problem with balancing requirements. It is a problem that typically arises in the context of the medium term management of a fleet of containers of multiple types, where container depots have to be selected, the assignment of customers to depots has to be established for each type of containers, and the interdepot container traffic has to be planned to account for differences in supplies and demands in various zones of the geographical territory served by a container shipping company.

The problem is modeled as a mixed integer program, which combines zero-one location variables (and constraints) and a multicommodity network flow structure (Crainic, Dejax and Delorme, 1989). Several exact and heuristic algorithms have been developed for this formulation, in particular a very efficient dual-based procedure (Crainic and Delorme, 1991). Given the fact that this problem is sufficiently close to general network design problems, this provides us with an ideal benchmark to critically assess the performance of Tabu Search based procedures for this class of problems, and should allow us to build a solid methodological platform for future developments.

This paper presents a Tabu Search algorithm for multicommodity location problems with balancing requirements, and is organized as follows. The first section briefly describes the problem and the mixed integer formulation. It also reviews some important

properties and the main algorithmic results obtained to date. Section 2 is dedicated to the detailed presentation of the Tabu Search procedure. Section 3 presents the computational results and analysis. We conclude by identifying future research directions.

1 The Problem and Model

Schematically, the problem is best described in the context of the management of a heterogeneous fleet of containers over a medium to long term planning horizon. Once a ship arrives at a port, the shipping company has to deliver loaded containers to their designated in-land destinations. These containers come in several types and sizes, and various transportation modes may be used to move them. Following their delivery and unloading by the importing customer, empty containers may be returned to the depot of the port that initiated the loaded movement, or may be repositioned to a different depot in prevision of future requests for empty containers. Indeed, the same, or other, customers are also requesting empty containers, of specific types, for subsequent shipping of their own products. Empty containers are delivered from depots, are loaded, and are subsequently moved, by land and maritime transportation means, to their final destinations. Additionally, regional unbalances in empty container availabilities and needs throughout the network require that empty containers be shipped between depots to "balance" the network. Interdepot transportation is assumed to be more efficient, per moved unit, than movements between depots and customers, due to the larger volumes of traffic which may be consolidated for transshipment over relatively long distances.

The general problem is therefore to locate the empty container depots in order to collect the supply of empty containers available at customers' sites and to satisfy the customer requests for empty containers, while minimizing the total operation cost: the cost of opening and operating the depots, plus the transportation costs between customers and depots and the costs generated by moving empty containers among depots to "balance" the network. In practical applications, this strategic/tactical logistics problem has to be repeatedly solved since, often, container shipping companies do not build their own depots, but rather use facilities available from other modes (e.g. ports and rail yards). It is then a question of deciding which facilities to use, according to the demand and the cost structure particular to the specific planning period. Dejax, Crainic and Delorme (1991), present a full description and discussion of the problem based on an actual application of planning the land operations of a maritime container shipping company.

Let C, D and P represent the sets of customers (cardinality m), candidate depots (cardinality n) and products (cardinality l), respectively. Crainic, Dejax and Delorme (1989) model this problem as:

$$\textit{(P) Minimize } Z = \sum_{j \in D} f_j y_j + \sum_{p \in P} \left\{ \sum_{i \in C} \sum_{j \in D} (c_{ijp} x_{ijp} + c_{jip} x_{jip}) + \sum_{j \in D} \sum_{k \in D} s_{jkp} w_{jkp} \right\}$$

$$\text{Subject to} \quad \begin{aligned} \sum_{j \in D} x_{ijp} &= O_{ip} \\ \sum_{j \in D} x_{jip} &= D_{ip} \end{aligned} \qquad \forall i \in C, p \in P \qquad (1)$$

$$\begin{aligned} x_{ijp} &\leq O_{ip} y_j \\ x_{jip} &\leq D_{ip} y_j \end{aligned} \qquad \forall i \in C, j \in D, p \in P \qquad (2)$$

$$\sum_{i \in C} x_{ijp} + \sum_{k \in D} w_{kjp} - \sum_{i \in C} x_{jip} - \sum_{k \in D} w_{jkp} = 0 \qquad \forall j \in D, p \in P \qquad (3)$$

$$\begin{aligned} x_{ijp} &\geq 0 \\ x_{jip} &\geq 0 \end{aligned} \qquad \forall i \in C, j \in D, p \in P \qquad (4)$$

$$w_{jkp} \geq 0 \qquad \forall j \in D, k \in D, p \in P \qquad (5)$$

$$y_j \in \{0,1\} \qquad \forall j \in D \qquad (6)$$

where

$y = (y_j)$	:	Opening vector for depots, $j \in D$; y_j equals 1 if depot j is open and zero otherwise;
x_{ijp}, x_{jip}	:	Flows (in both directions) of product p, $p \in P$, between customer i, $i \in C$, and depot j, $j \in D$;
w_{jkp}, w_{kjp}	:	Flows of product p, $p \in P$, between depots j and k, $j \in D$, $k \in D$;
f_j	:	"Fixed" cost of depot j, $j \in D$;
c_{ijp}, c_{jip}	:	Unit transportation costs for product p, $p \in P$, between customer i, $i \in C$, and depot j, $j \in D$;
s_{jkp}, s_{kjp}	:	Unit transportation costs for product p, $p \in P$, between depots j and k, $j \in D$, $k \in D$;
O_{ip}	:	Supply at node (customer) i, $i \in C$, of product p, $p \in P$;
D_{ip}	:	Demand at node (customer) i, $i \in C$, of product p, $p \in P$;

Constraints (1) ensure that demand and supply requirements are met. Constraints (2) forbid exchanges between customers and closed depots (analogous constraints for interdepot movements have been shown to be redundant if the applicable transportation costs satisfy the triangle inequality; they can thus be dropped). Constraints (3) correspond to the interdepot balancing requirements. Figure 1 illustrates the structure of the problem.

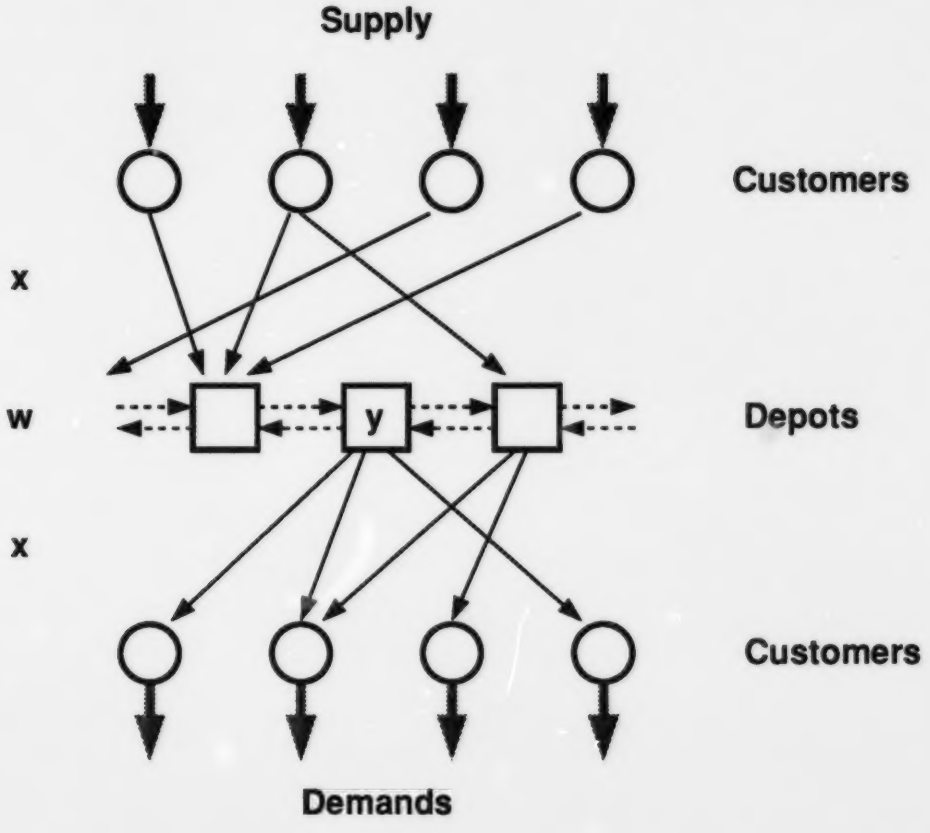

Figure 1. Network flow structure for the problem

Several algorithmic developments have already been performed for this problem. The exploration of classical heuristic approaches (greedy, interchange, etc.) has been initiated with encouraging success (Guo, 1990). Lagrangian relaxation approaches have proved quite efficient for moderate size problems, but their performance seems to deteriorate with the number of commodities (Guo, 1990; Guo et al., 1989). Research is continuing in this direction, as well as in the area of primal/dual approaches (Tétreault, 1990).

Crainic, Delorme and Dejax (1991) have proposed a branch-and-bound algorithm to solve exactly the problem. In particular, they have shown that classical results for the simple plant location problem do not extend to this problem, and have developed and analyzed specific branching and bounding rules. Recently, Crainic and Delorme (1991) have presented an extremely efficient dual-ascent heuristic procedure. In fact, for now, this is the most performant procedure for the multicommodity location problem with balancing requirements. Yet, most solutions display optimality gaps larger than 1% and, indeed, the procedure finds local optima for several problems. Hence, improvements are still possible and we aim at achieving some through the application of Tabu Search.

An important characteristic of the present formulation consists in the network structure that underlies it and which is apparent in Figure 1. This property is extensively used in the various algorithmic developments. The particular form that we will use to develop the Tabu Search method concerns the formulation $(P(\overline{y}))$ which results when the location variables y_j are fixed to some values $\overline{y}_j$, $j \in D$:

$$(P(\overline{y}))\ \ Minimize\ Z(P(\overline{y})) = \sum_{p \in P} \left(\sum_{i \in C} \sum_{j \in \overline{D}} (c_{ijp} x_{ijp} + c_{jip} x_{jip}) + \sum_{j \in \overline{D}} \sum_{k \in \overline{D}} s_{jkp}\, w_{jkp} \right)$$

$$\text{Subject to} \quad \sum_{j \in \overline{D}} x_{ijp} = O_{ip} \qquad i \in C, p \in P$$

$$\sum_{j \in \overline{D}} x_{jip} = D_{ip}$$

$$\sum_{i \in C} x_{ijp} + \sum_{k \in \overline{D}} w_{kjp} - \sum_{i \in C} x_{jip} - \sum_{k \in \overline{D}} w_{jkp} = 0 \qquad j \in \overline{D}, p \in P$$

$$x_{ijp}, x_{jip} \geq 0 \qquad i \in C, j \in \overline{D}, p \in P$$

$$w_{jkp} \geq 0 \qquad j \in \overline{D}, k \in \overline{D}, p \in P$$

where $\overline{D}$ represents the set of all open $(i.e., \overline{y}_j = 1)$ depots. We may then show that:

Property 1.

$(P(\overline{y}))$ is an uncapacitated multicommodity minimum cost network flow problem.

2 The Tabu Search Procedure

2.1 The search space

The Tabu Search procedure that we propose to solve our problem exploits Property 1 of the previous section. More precisely, since it is possible to obtain "optimal" values of the continuous flow variables x and w corresponding to any given vector of location variables $\overline{y}$ by solving a series of minimum cost flow problems, the search procedure can be restricted to the space of the location variables. In the following, when we refer to a "solution", this should thus be interpreted as an assignment of 0's and 1's to the location variables. The objective function value associated with a solution $\overline{y}$ is then $Z(\overline{y}) = \sum_{j \in D} f_j \ \overline{y}_j + Z(P(\overline{y}))$.

Since most customers are linked to only a subset of the potential depot locations, there are customers who, in some solutions, do not have access to an open depot, thus making those solutions *infeasible*. In an early implementation of the algorithm, we tried restricting the search to feasible solutions, but this proved impractical, since the search space was too constrained. To circumvent this difficulty, an *artificial depot* was added. This depot (which is always considered to be open) is linked to all customers (but not to the other depots) by very high cost arcs. These costs are such as to guarantee that the artificial arcs will bear no flow, unless the solution is infeasible. This provides us with

an easily implemented test regarding the feasibility of solutions: a solution is feasible if and only if the total flow on the artificial arcs is zero. As the search proceeds, we keep track of the feasibility of the solutions encountered to make sure that the best solution discovered is indeed feasible and to adjust some control parameters of the procedure as will be explained later on.

2.2 Neighbourhood structure

The basic iterative step of any Tabu Search procedure involves moving from a current point (solution) in the search space to one of its "neighbours" according to some suitably defined *neighbourhood structure*. With regards to the problem at hand, a "natural" neighbourhood structure can be defined by considering moves in which a single y_j variable is modified (complemented), i.e. the status of a single depot is changed (from open to closed or vice-versa). Our search procedure indeed allows for such moves: moves in which a depot is opened are called *add moves*, and moves in which a depot is closed are called *drop moves*. However, early on in the development of the procedure, it was found out that add/drop moves suffered from several disadvantages, especially when fixed costs of depots are large relative to the other costs. First, with them, objective function values tend to vary wildly from one solution to the next; second, since in most problems, the good solutions tend to have the same number of open depots, add/drop moves do not allow for the most efficient exploration of the solution space in the latter stages of the search.

A third type of moves was therefore defined: *swap moves* in which we simultaneously open a depot and close another one (this amounts to performing at once an add and a

drop moves). Since the number of possible swaps at any given time is much larger than the number of possible add/drop moves, and since swaps between depots that are far apart are generally unprofitable, it was decided to restrict swaps to pairs of depots which are physically close: for each open depot j, we only consider the closed depot k which is the "closest" (as measured by the balancing flow costs, i.e. $\sum_p s_{jkp}$). A nice feature of swap moves is that they allow for an efficient implementation of search intensification strategies by keeping the number of open depots unchanged.

2.3 The search strategy

The search strategy is organized around two complementary objectives: 1) determining the optimal number of open depots, 2) for a fixed cardinality of the set of open depots, finding the best possible configuration of open depots. The first objective is accomplished by performing sequences of add and drop moves (these are called *add/drop sequences*) which allow for the examination of open depot sets of various sizes, while the second is pursued by consecutively executing several swap moves (thus creating *swap sequences*). Swap sequences are further divided into *normal swap sequences*, in which the best candidate move is always implemented, and *strict swap sequences*, in which swaps are executed only if they result in an improvement of the current solution.

The basic step of the search procedure consists in an add/drop sequence followed by a normal swap sequence which is initiated from the best solution found in the add/drop sequence. This combined add/drop-normal swap sequence yields a best "local" solution which is either feasible or infeasible. If it is feasible, we immediately perform *search intensification* by executing a strict swap sequence starting from the local solution. If

the local solution is infeasible, another add/drop-normal swap sequence, starting from the last solution obtained in the previous normal swap sequence, is initiated after adjustment of the search control parameters. This process is repeated until a feasible local solution is encountered.

A portion of the search beginning with an add/drop sequence and terminating with a strict swap is called an *inner loop*; N of these inner loops (where N is an input parameter) followed by a *search diversification* step (described later on) make up an *outer loop*. The overall search procedure starts from an initial solution and performs a sequence of outer loops until some termination criterion is met. In the current implementation, this termination criterion is simply the total number of iterations since the beginning of the search.

The general structure of the search procedure is summarized in Figure 2.

2.4 Initial solution

The initial solution from which the search is to proceed, can be determined in several ways. The simplest one corresponds to opening all depots, i.e. setting $y_j = 1$, $\forall j \in D$. Such an initial solution has, in general, the disadvantage of being "far" from good solutions, since, in most cases, the number of open depots in these is much smaller than $|D|$. An immediate consequence of this situation is that the procedure has to close several depots before starting to explore interesting areas of the search space.

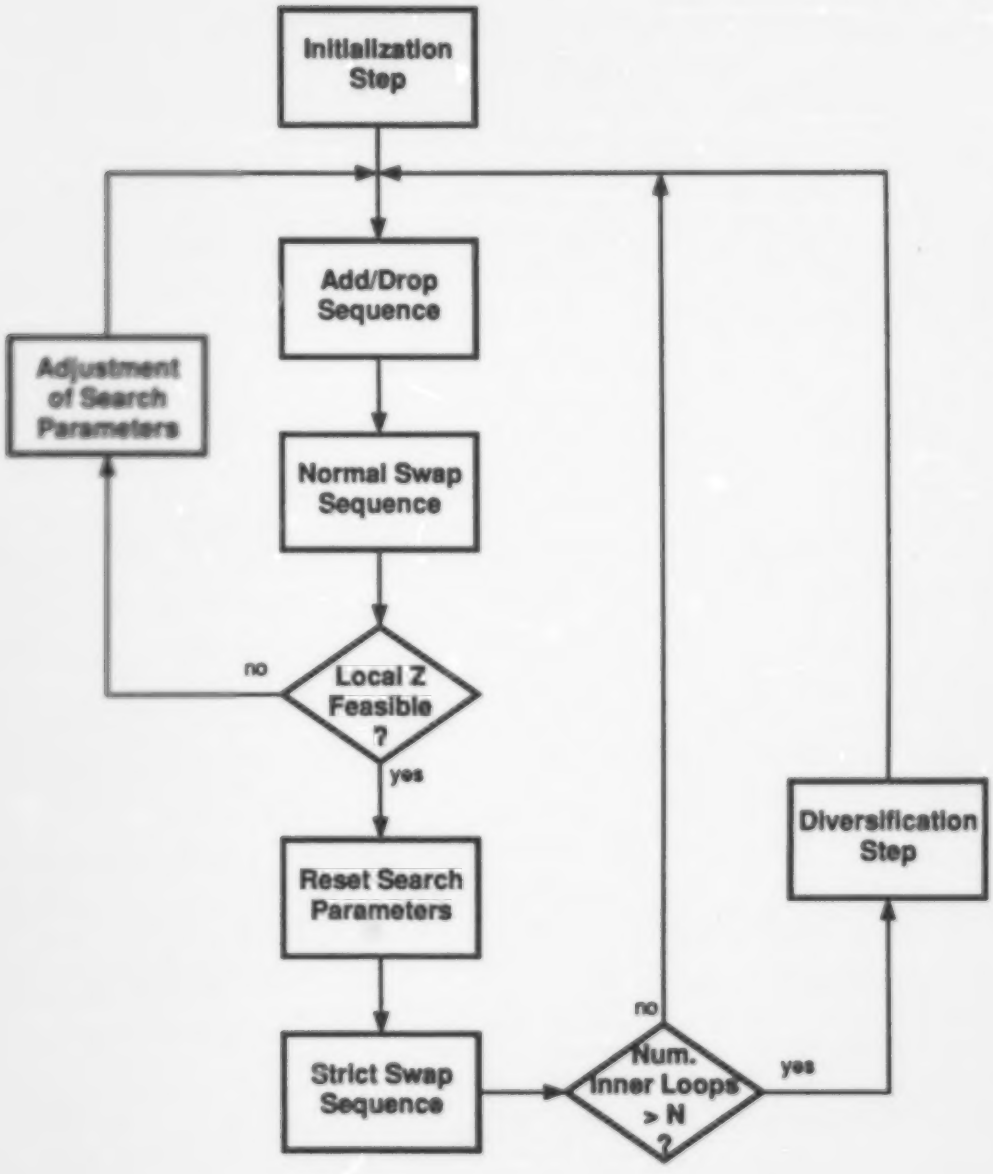

Figure 2: Structure of the search procedure

An alternative method for selecting an initial solution consists of first determining an estimate δ (not necessarily very precise) of the number of open depots in good solutions and then opening the δ depots with lowest fixed costs. This is the approach we have used in our implementation.

2.5 Implementation of add/drop and swap sequences

The two types of sequences share a number of common features. First, at each iteration of both, only a random sample of possible moves is considered, in order to reduce the computational burden. These samples are not of fixed sizes, but are constructed using specified *selection probabilities*: for each possible move under the current sequence, a uniform (0,1) random number is drawn; the move is included in the sample if this number falls below the corresponding probability. An interesting consequence of this sample building scheme with respect to add/drop sequences is that it provides a natural adjustment mechanism of the proportion of add and drop moves making up the sample: when most depots are open, the sample is mainly made up of drop moves, while the opposite is true when there are only a few open depots. We noted, however, that this procedure could, under certain circumstances, lead to situations where a sample would contain no, or almost no, moves of a given type. A minimum value of 4 was thus imposed on the number of potential add and drop moves to be included in a sample of add/drop sequences (if less than 4 moves of a given type are possible, all of them are considered).

At the beginning of the algorithm, initial values must be provided for the selection probabilities of add, drop and swap moves. These selection probabilities do not however remain fixed during the whole search. Indeed, these constitute an important control

mechanism with regards to feasible/infeasible solutions. Since low selection probabilities tend to constrain the search, we increase the current probability values whenever a combined add/drop normal swap sequence fails to produce a feasible solution. The updating schedule is as follows: add/drop selection probabilities are increased by 0.08 and swap probabilities by 0.05 (up to, naturally, a maximum value of 1). To help in the search for feasible solutions, the cost values of artificial arcs are also multiplied by 2 when selection probabilities are increased. All these search control parameters are reset at their original values at the beginning of each outer loop.

Another feature shared by add/drop and swap sequences is that their length is not known beforehand: each sequence continues until a number of iterations has been performed without producing an improvement in the value of the local optimum. Add/drop and swap sequences however use distinct termination criteria which are specified as input parameters. It is possible to significantly alter the behaviour of the search by varying these parameters, as will be discussed in the section on computational results.

Add/drop and swap sequences both use *short-term memory* tabu lists. For add/drop sequences, there is one list in which are recorded the T_1 last depots added or dropped from the solution. The reverse moves are forbidden as long as the depots remain on the list. For swap sequences, a list, which records the most recently performed swaps as pairs of depots, is maintained. It prevents a reversal or a repetition of the T_2 last moves. It is necessary to forbid the repetition of swaps, because cycling can occur when only reversals are prohibited. For instance, a sequence such as *{(close* j_1*, open* j_2*), (close* j_2*, open* j_3*), (close* j_3*, open* j_1*), (close* j_1*, open* j_2*), ...}* would be legal in that case.

All short-term tabu lists are emptied at the beginning of swap sequences since these start at the best local solution (thus invalidating the recent history of the search). At the end of swap sequences, however, the information recorded in the swap tabu list is transfered to the add/drop list (each swap is then interpreted as the combination of an add and a drop move).

2.6 Evaluation of neighbours

As was mentioned earlier, the computation of the exact value of the objective function associated with a given solution $\overline{y}$ involves solving the multicommodity flow problem $P(\overline{y})$, or equivalently a series of single commodity minimum cost flow problem. Even if an efficient network flow code such as RNET (Grigoriadis and Hsu, 1979) is used, it is extremely time-consuming to evaluate all candidate moves exactly. To alleviate this problem, we have decided to compute estimates, rather than exact values, of the objective function for moves. The basic principle underlying these estimates is that flow patterns should not vary very much when a single depot is opened or closed.

Add moves

Let $\tilde{y}$ be a candidate solution obtained from the current solution $\overline{y}$ by opening depot $\tilde{j}$. To estimate the difference $P(\tilde{y}) - P(\overline{y})$, we re-assign to depot $\tilde{j}$ every customer which is located closer (with respect to cost) to $\tilde{j}$ that to the depot to which it is currently assigned under $\overline{y}$. This will have two consequences: a decrease in customer-to-depot (and depot-to-customer) flow costs, which is easily accounted for, and a modification of the *net flow balance* at each depot, for each product, which will result in different *flow balancing costs*. Computing exactly the new flow balancing costs would involve

solving for each product a classical transportation problem and is therefore not attractive. Instead, we use the following formula:

$$C_b(\tilde{y}) \;=\; \sum_{p \in P} \overline{\xi}^p_\beta \left(\sum_{j \in \tilde{D}} \left| V^p_j \right| \right) \;/\; 2$$

where

$C_b(\tilde{y})$	is the estimate of the balancing flow costs under $\tilde{y}$,
$\tilde{D}$	is the set of open depots in $\tilde{y}$,
β	is the number of open depots in $\tilde{y}$,
V^p_j	is the flow balance of product p at depot j,

and

$\overline{\xi}^p_\beta$	is an estimate of the unit balancing flow cost for product p for configurations with β open depots.

The estimate $\overline{\xi}^p_\beta$ can be derived from previous solutions through an exponential smoothing procedure. Let $\overline{w}$ be the vector of balancing flows associated with a solution $\overline{y}$ with β open depots. Then the unit balancing flow costs for product p is given by

$$\xi^p_\beta(\overline{y}) \;=\; \left(\sum_{j \in \overline{D}} \sum_{k \in \overline{D}} s_{jkp}\, w_{jkp} \right) \Big/ \sum_{j \in \overline{D}} \sum_{k \in \overline{D}} w_{jkp}.$$

Initially, all $\overline{\xi}^p_\beta$ are set equal to zero. The first time that a solution $\overline{y}$ with β open depots is visited, $\overline{\xi}^p_\beta$ is reset to $\xi^p_\beta(\overline{y})$, for every p. Later in the search, when other solutions $\overline{y}$ with β open depots are encountered, $\overline{\xi}^p_\beta$ is updated using the classical rules of exponential smoothing: $\overline{\xi}^p_\beta := \alpha\, \xi^p_\beta(\overline{y}) + (1 - \alpha)\, \overline{\xi}^p_\beta$ where α is the smoothing constant $(0 < \alpha < 1)$, which is an input parameter.

Drop moves

The procedure for evaluating drop moves is quite similar to the previous one, except that we simply reassign to the closest open depot the customers which were allocated to the depot which is to be closed if the move is executed. Note however that, in this case, customer-to-depot flow costs may increase or decrease depending on the exact configuration of the solution.

Swap moves

Every swap move is evaluated as the combination of a drop move followed by an add move using the procedures defined above.

Adjustment of balancing flow costs

During preliminary testing of the heuristic, it was found out that our estimates of balancing flow costs were sometimes too optimistic (i.e. too low). An *adjustment factor* η, that varies with problem size, is thus used to multiply the balancing flow costs before computing the value of the objective function estimates.

2.7 Search diversification

After performing N inner loops, the search procedure is re-directed to previously unexplored portions of the search space by performing a *search diversification step*. This step uses a *long-term memory* in which we record the number of times that the status of each depot has been modified (changed from open to closed or vice-versa).

The search diversification step is always performed starting from the best global solution found up to this point in the search. Based upon the values stored in the long-term

memory, the γ depots with the lowest activity counts (number of status modifications) are selected. The status of each of these depots is then changed, providing thus a new solution which differs from the best one in exactly γ components. To prevent too quick a reversal of these depots to their previous status, they are added to the short-term memory.

Considering the fact that values in the long-term memory tend to evolve slowly, it was felt necessary to provide a *diversification memory* in order to record the last set of depots that were selected for diversification. These are then excluded from consideration in the next diversification step to prevent repetitive patterns.

3 Computational Results

Extensive computational testing was performed to assess the impact of parameter settings on the quality of the solutions produced by our method and to determine whether Tabu Search is a competitive approach for this problem when compared to more traditional heuristics such as dual ascent.

3.1 The problem set

The set of benchmark problems for these tests was extracted from a much larger set of randomly generated problems that had been created previously to evaluate other solution approaches. 20 problems made up this set : 12 of these are "medium-size" problems (numbered M1 to M12) with about 25 potential depot locations and 125 customers, while the remaining 8 are "large-size" problems (numbered L1 to L8) with about 44 depot sites and 220 customers. The number of different container types ("products") ranges from 1 to 6 for the medium problems and from 1 to 2 for the large ones.

Each group of problems can be further broken into two subgroups : one with low fixed-to-variable cost ratios (M1 to M6 and L1 to L4) and the other with high ratios (M7 to M12 and L5 to L8); in fact, the problems with the high ratios were obtained from the ones with the low ratios by multiplying the depot fixed costs by 10 (for instance, problem M7 is the same as M1 except for fixed costs). The fixed-to-variable cost ratio is an important attribute of the problems, since it was known beforehand that problems with high ratios display much larger dual gaps than those with low ratios.

The last attribute of the problems is the number of "regions" that they cover. This should be interpreted as a measure of the level of imbalance in the network supply and demand for the various products : the larger this number, the higher will balancing flows between depots be. This attribute does not seem to have a large impact on the quality of the solutions, but it allows one to create problems that are not too similar.

The problem characteristics are summarized in Table 1. Further details on the generation procedure for these problems can be found in Crainic, Delorme and Dejax (1990).

3.2 Experimentation with parameter settings

The experimentation on parameter settings was conducted in two phases. The first phase (preliminary testing) consisted in solving subsets of the problem set with various settings of all parameters to determine whether some of these could be fixed once and for all, either because the solution quality was insensitive to them or because some settings were obviously superior to others. At the end of this phase, it was decided to fix, for all

Problem	Customers	Depot Candidates	Regions	Products	Level of Fixed Costs
M1	125	25	1	1	Low
M2	125	25	1	4	Low
M3	124	26	2	3	Low
M4	125	25	2	6	Low
M5	124	26	4	1	Low
M6	124	26	4	4	Low
M7	125	25	1	1	High
M8	125	25	1	4	High
M9	124	26	2	3	High
M10	125	25	2	6	High
M11	124	26	4	1	High
M12	124	26	4	4	High
L1	219	44	1	1	Low
L2	219	44	1	2	Low
L3	220	43	3	1	Low
L4	219	44	3	2	Low
L5	219	44	1	1	High
L6	219	44	1	2	High
L7	220	43	3	1	High
L8	219	44	3	2	High

Table 1 : Test problem characteristics

subsequent runs, the number of iterations to 300 (longer runs did seldom produce better solutions while requiring proportionally larger CPU times), the number of inner loops in an outer loop to 2, the exponential smoothing constant for unit balancing flow cost estimates α to 0.3, and the balancing flow cost adjustment factor η to 1 for medium problems and 1.1 for large ones. As for the number of open depots in initial solutions, it was set to 5 for medium-sized problems with high fixed costs (M7–M12), to 10 for medium problems with low fixed costs (M1–M6) and for large problems with high fixed

costs (L5–L8), and to 15 for large problems with low fixed costs (L1–L4).

The second phase of parameter setting experimentation (systematic testing) involved solving the whole problem set repeatedly, while varying one or two parameters and keeping the others fixed. We now report on the results of this phase.

Tabu list length and number of iterations without improvement

The first obvious candidates for systematic testing are the lengths of the two short-term tabu lists. However, considering the fact that there might be a strong interaction between the length of a tabu list and the number of iterations without improvement that are allowed in a given sequence (since tabu lists are an anti-cycling device), it was deemed necessary to vary, for each type of sequence, these two parameters together.

Three lengths were considered for the tabu lists : 2, 5 and 8 (preliminary testing had shown that larger list sizes were inefficient). For each list size, three values of the number of iterations without improvement (called "repetitions" in the following and in the tables) were examined; these correspond to ratios of repetitions over list size of 3/2, 2 (4/2) and 5/2 (for a list size of 5, repetitions thus takes on the values 8, 10 and 13.

The settings of the remaining parameters were for each experiment:

- size of the other short-term tabu list, 5;
- repetitions for the other type of sequence, 8;
- sampling probability for all sequences, 0.3;
- diversification amplitude γ, 6 for medium problems and 12 for large problems.

The results of these runs are summarized in Table 2 for add/drop sequences and in Table 3 for swap sequences. Looking at these tables, it may fairly easily be concluded that :

- for add/drop sequences, two combinations of tabu list length and repetitions value yield the best results for both medium and large problems: (2,5) and (5,10);
- for swap sequences, the pair (2,3) is clearly superior to all others.

Ratio of repetitions over list size	Tabu list size					
	Size = 2		Size = 5		Size = 8	
	M	L	M	L	M	L
3/2	5	1	4	2	3	0
4/2	7	2	7	3	5	2
5/2	7	3	5	2	6	1

Table 2 : Number of instances where best overall solution was found with respect to ADD/DROP tabu list size and repetitions over list size ratio for Medium and Large problems

Given that the add/drop tests had been conducted with a swap tabu list of size 5 and swap repetitions value of 8, we were curious to see how the two best add/drop combinations would perform when these values would be changed respectively to 2 and 3. The 20 problems were thus solved again for these settings, leaving all remaining parameters unchanged. The results obtained are displayed in Table 4.

Ratio of repetitions over list size	Tabu list size					
	Size = 2		Size = 5		Size = 8	
	M	L	M	L	M	L
3/2	7	4	5	2	6	1
4/2	7	1	3	1	5	1
5/2	6	1	6	2	4	2

Table 3 : Number of instances where best overall solution was found with respect to SWAP tabu list size and repetitions over list size ratio for Medium and Large problems

Analyzing the results of Table 4, it can be seen that the shorter add/drop tabu lists yield better results (lower gaps with respect to a lower bound provided by a dual ascent procedure) for 10 problems out of 20, while the longer list is better only for 6 problems (out of the 4 other problems for which the two lists sizes are equivalent, 3 were in fact solved to optimality). From this, we may conclude that it is indeed better to use the shorter list, since it requires less computing effort and it produces on the average better solutions.

The results of these experiments may seem surprising at first sight. Indeed, how can it be that the best results are achieved with short-term tabu lists of length as small as 2? To understand this somewhat counter-intuitive result, one must go back to the overall organization of the search procedure. Shorter list lengths together with low repetitions

Problem	2/5-2/3	5/10-2/3
M1	0.681	0.663
M2	0.906	0.653
M3	0.700	1.136
M4	0.689	0.801
M5	0.553	0.553
M6	0.709	1.280
M7	1.907	0.789
M8	1.632	1.632
M9	0.777	0.777
M10	1.150	0.673
M11	3.240	3.240
M12	2.067	3.152
L1	0.296	0.726
L2	2.770	2.312
L3	0.342	0.939
L4	2.262	1.918
L5	3.418	6.340
L6	1.755	4.322
L7	1.031	2.313
L8	2.985	7.162

Table 4 : Relative gaps (%) with respect to the dual lower bound for the two best list size/repetitions settings

values imply that each sequence in the search is fairly short and that the search process thus alternates often between the two types of sequences. This, in turn, may help directing the search to the most promising regions of the solution space. Finally, another factor must be taken into account. Since the number of inner loops per outer loop is fixed, shorter sequences result in shorter inner loops and therefore in shorter outer loops, which means that diversification is performed more often. This by itself allows for the exploration of a larger portion of the solution space.

Sampling probabilities

The next series of experiments involved changing the sampling probabilities for both add/drop and swap moves. These were modified simultaneously, since it was felt that both probabilities would have a similar impact on the performance of the method. Three different probability levels were considered: 0.3, 0.5 and 0.7. The best values for tabu list lengths and repetitions were used and all other parameters were left unchanged. The results obtained are displayed in Table 5.

These results show that, in general, sampling probabilities have a very small impact on the quality of the solution. However, this statement must be qualified. Preliminary testing with very low sampling probabilities such as 0.1 had yielded notingly inferior solutions. It would thus be more correct to state that 0.3 is a sampling probability value large enough for the method to perform adequately and that there are no significant gains to be made by using larger values (this only increases CPU time needlessly).

Diversification amplitude

The last series of experiments aimed at evaluating the influence that changing the

amplitude of the diversification step would have. To do so, we tested values of γ corresponding to 1/2 and 3/2 of its initial setting for both medium and large problems (i.e. γ = 3 and 9 and γ = 6 and 18 respectively), while setting all other parameters to the values used in the leftmost column of Table 5. The results obtained are reported in Table 6.

Problem	0.3	0.5	0.7
M1	0.681	0.663	0.663
M2	0.906	0.619	0.879
M3	0.700	1.276	0.844
M4	0.689	0.567	0.751
M5	0.553	0.553	0.553
M6	0.709	0.938	0.572
M7	1.907	0.789	0.789
M8	1.632	1.813	1.632
M9	0.777	0.777	0.777
M10	1.150	0.999	0.673
M11	3.240	3.240	3.240
M12	2.067	2.339	3.848
L1	0.296	0.296	0.296
L2	2.770	1.977	2.133
L3	0.342	0.401	0.330
L4	2.262	2.028	1.821
L5	3.418	3.332	3.332
L6	1.755	2.222	3.749
L7	1.031	1.860	1.860
L8	2.985	3.856	5.193

Table 5 : Influence of sampling probabilities on relative gap (%)

Problem	γ=3	γ=6	γ=9
M1	0.663	0.681	0.663
M2	1.039	0.906	0.854
M3	1.580	0.700	0.966
M4	0.568	0.689	0.989
M5	0.553	0.553	0.553
M6	0.286	0.709	0.938
M7	0.789	1.907	1.907
M8	1.632	1.632	1.632
M9	0.777	0.777	1.145
M10	0.995	1.150	0.734
M11	3.240	3.240	3.240
M12	2.067	2.067	5.762
Problem	**γ=6**	**γ=12**	**γ=18**
L1	0.296	0.296	0.296
L2	1.727	2.770	2.519
L3	0.841	0.342	0.330
L4	1.980	2.262	2.262
L5	5.602	3.418	3.332
L6	3.449	1.755	0.895
L7	1.862	1.031	1.862
L8	4.117	2.985	5.160

Table 6 : Influence of diversification amplitude on relative gap (%)

Analyzing the results of Table 6, one may observe that for medium problems the smallest value of γ (γ = 3) yields the best results. For large problems, the settings γ = 12 and γ = 18 both provide good results on the average. However, if one is concerned with the robustness of the method, the value of γ to be preferred is 12, since large gaps

(> 5%) are sometimes obtained with $\gamma = 18$ (e.g. problem L8).

3.3 Comparison with dual ascent

In order to assess the competitiveness of Tabu Search for this type of problem, we compared our results with those obtained by the dual ascent heuristic developed by Crainic and Delorme (1991). This procedure has been proven to be very effective on a large set of test problems (which included our benchmark problems) and was therefore an obvious candidate for comparison.

Table 7 displays the results obtained by the two approaches. As in previous tables, these are reported in terms of the relative gap with respect to the lower bound provided by the dual procedure. The results of the Tabu Search heuristic are listed under the heading "Gap (%)" and those of the dual ascent procedure under "DUAL Gap (%)". In addition, the iteration at which the best solution was found and the CPU time (on a SUN SPARCstation 2) elapsed at that moment are reported in columns "Best Z Iteration" and "Best Z Time" respectively. Finally, the total CPU time used by the Tabu Search procedure to perform the 300 iterations it was allowed is listed under "Total Time (300 iterations)". Optimal solutions are indicated by asterisks ("*"). The parameter settings used were those identified in the previous tests as being the best overall (in terms of solution quality and robustness), that is: add/drop tabu list length = 2, add/drop repetitions = 5, swap tabu list length = 2, swap repetitions = 3, sampling probabilities = 0.3, γ = 3 for medium problems and 12 for large ones.

Problem	Gap (%)	Best Z Iteration	Best Z Time	Total Time (300 Iterations)	DUAL Gap (%)
M1	0.663*	166	102	186	0.663*
M2	1.039	266	704	790	0.296*
M3	1.580	166	296	550	1.126
M4	0.568	108	441	1149	0.271
M5	0.553*	33	22	186	1.849
M6	0.286*	237	627	787	0.652
M7	0.789*	13	6	169	6.935
M8	1.632*	10	19	690	3.009
M9	0.777*	252	395	478	0.777*
M10	0.995	163	543	1007	2.068
M11	3.240*	13	7	175	4.355
M12	2.067	214	482	685	2.412
L1	0.296*	32	37	310	0.573
L2	2.770	92	236	762	0.980
L3	0.342	175	213	356	1.337
L4	2.262	297	711	715	0.834
L5	3.418	203	211	302	24.376
L6	1.755	286	643	670	0.895*
L7	1.031*	241	270	335	2.313
L8	2.985	229	486	620	2.940

Table 7 : Computational results with the chosen parameter settings (A/D list = 2, A/D reps. = 5, Swap list = 2, Swap reps. = 3, sampling = 0.3, $\gamma = 3$ for Medium problems and 12 for Large problems) and corresponding DUAL gaps (%)

From Table 7, it may be concluded that Tabu Search is indeed a competitive solution technique for location/allocation problems with balancing flows: 1) Tabu Search finds the optimal solution for 9 problems out of 20, while dual ascent is successful in only 4 instances; 2) Tabu Search yields a better solution than dual ascent for 11 of the problems (and the two methods find the same solutions in two cases); 3) for Tabu Search, the largest gaps are inferior to 4%, while values as large as 24% occur with dual ascent, which indicates that our method is much more robust.

Conclusion

We have presented a Tabu Search heuristic for solving location/allocation problems with balancing requirements. Extensive computational testing was performed on a set of benchmark problems. These tests showed that Tabu Search finds excellent, if not optimal, solutions to all problems in a reasonable amount of CPU time. Comparisons with a very effective dual procedure, especially customized for this class of problems, were quite favourable.

Since the proposed approach did not exploit the particular structure of the problem, it should be relatively easy to adapt it for solving similar problems complicated by additional constraints (e.g. budget constraints, capacity constraints, etc.). The results obtained are also very encouraging with regards to the perspectives of application of Tabu Search to mixed integer programming problems, in particular to those with an underlying network flow structure.

Acknowledgements

This work was partially funded by the Canadian Natural Sciences and Engineering Research Council and by the Fonds F.C.A.R. of the Province of Quebec. This support is gratefully acknowledged.

REFERENCES

Balakrishnan A., Magnanti T.L. and Wong R.T. (1989), *A Dual-Ascent Procedure for Large-Scale Uncapacitated Network Design*, **Operations Research** 37(5), 716–740.

Crainic T.G., Dejax P.J. and Delorme L. (1989), *Models for Multimode Multicommodity Location Problems with Inderdepot Balancing Requirements*, **Annals of Operations Research** 18, 279–302.

Crainic T.G., Delorme L. and Dejax P.J. (1990), *A Branch-and-Bound Method for Multicommodity Location with Balancing Requirements*, forthcoming in **European Journal of Operational Research.**

Crainic T.G. and Delorme L. (1991), *Dual-Ascent Procedures for Multicommodity Location/Allocation Problems with Balancing Requirements*, to appear in **Transportation Science.**

Dejax P.J., Crainic T.G. and Delorme L. (1991), *Strategic/Tactical Planning of Container Land Transportation Systems*, forthcoming in **European Journal of Operations Research.**

de Werra D. and Hertz A. (1989), *Tabu Search Techniques - A Tutorial and Application to Neural Networks*, **OR Spektrum** 11, 131–141.

Gendreau M., Hertz A. and Laporte, G. (1991), *A Tabu Search Heuristic for the Vehicle Routing Problem*, Publication #777, Centre de recherche sur les transports, Université de Montréal.

Glover F. (1986), *Future Paths for Integer Programming and Links to Artificial Intelligence*, **Computers and Operations Research** 13, 533–549.

Glover F. (1989), *Tabu Search - Part I*, **ORSA Journal on Computing** 1(3), 190–206.

Glover F. (1990), *Tabu Search - Part II*, **ORSA Journal on Computing** 2(1), 4–32.

Grigoriadis M.D. and Hsu T. (1979), *RNET - The Rutgers Minimum Cost Network Flow Subroutines*, Rutgers University, New Brunswick, New Jersey.

Guo Z.-Y. (1990), *Résolution heuristique et optimale du problème de localisation de dépôts avec équilibrage*, Ph.D. Thesis, Département de Génie Industriel, Ecole Centrale, Paris.

Guo Z.-Y., Dejax P., Crainic T.G. and Delorme L. (1989), *Résolution heuristique du problème de localisation-distribution avec équilibrage*, **Logistique/Logistics: Production, Distribution, Transport/Manufacturing, Distribution, Transportation,** Proceedings of *Conference on the Practice and Theory of Operations Management* and *4es journées francophones sur la Logistique et les Transports*, AFCET, 87–95.

Skorin-Kapov, J. (1990), *Tabu Search Applied to the Quadratic Assignment Problem*, **ORSA Journal on Computing** 2, 33–45.

Tétreault N. (1990), *Une approche de restriction basée sur une méthode primale-duale pour résoudre le problème de localisation-distribution avec échanges interdépôts*, M.Sc. Thesis, Publication #681, Centre de recherche sur les transports, Université de Montréal.

Widmer M. and Hertz A. (1989), *A New Heuristic for Solving the Flow Shop Sequencing Problem*, **European Journal of Operational Research** 41, 186–193.